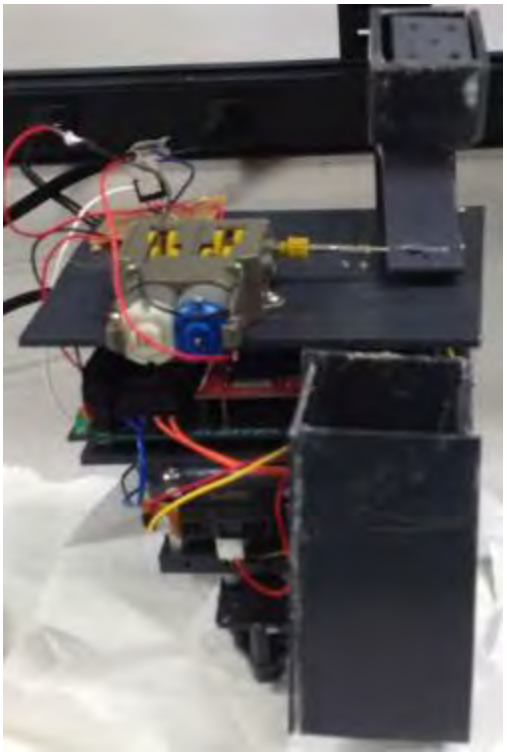
Search and Rescue Robot

(SRR)



Written by:

Vivek Jain & Pranav Subramanian

Research Advisor: Professor Joseph S. Miles

Date: August 8th, 2014

Stevens Institute of Technology

Abstract

Autonomous robots can perform a variety of operations without human guidance in a particular environment. These robots can evaluate their surroundings and receive data from sensors via hardware and process it using software. Autonomous robots are most prominently used in factories, industries, space exploration, and UAVs'. Being one of the leading technologies in today's industry, these robots can carry out a plethora of civilian purposes.

The goal of the project was to construct a Search and Rescue Robot capable of dropping a cube 12 inches from an infrared beacon while avoiding obstacles in its path and adapting to a dynamic environment. The robot had to utilize various pieces of equipment to accomplish its task. Such equipment included various sensors to detect distance and light and a drop-off mechanism consisting of an arm and funnel. Other than rescue operations, this project demonstrated that such a design has great potential for applications in space as well. Some include dropping a marker at a specific location on different planets for future explorers.

TABLE OF CONTENTS

Introduction

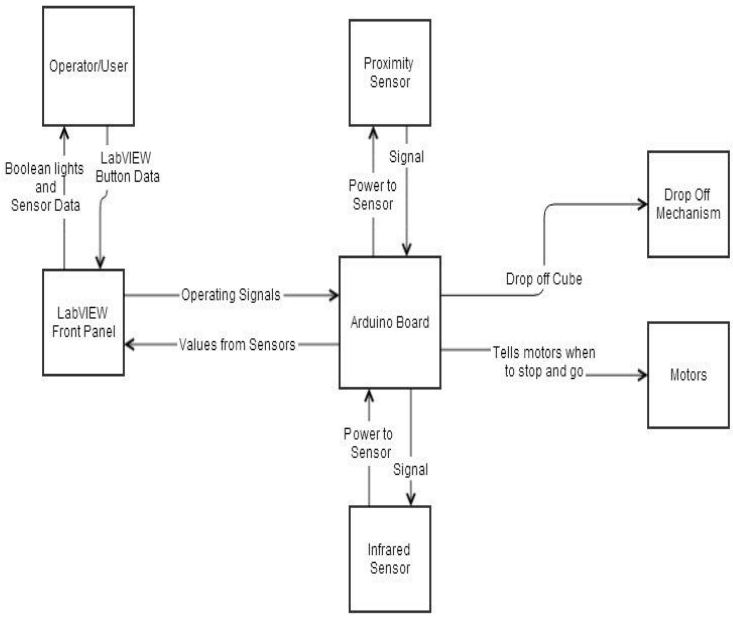
The objective of this project was to construct a robot that could drop a cube 12 inches from an infrared beacon. The robot accomplished this task through a plethora of equipment, such as sensors and a drop-off mechanism. It primarily used an infrared sensor and a proximity sensor to accomplish this task. The infrared sensor ensured the detection of the infrared beacon, and the proximity sensor calculated how far the robot was from the beacon. The robot was powered by three motors and an Arduino microcontroller.

The robot was controlled by a Logitech 3D Extreme Pro Joystick. This joystick was calibrated using an X, Y, Z, and Z rotational axis. The software used to program the robot was the 2011 LabVIEW Program. This program allowed multiple "sub-programs" to be constructed for the different aspects of the robot that needed to be programmed.

Working on this project demonstrated that this robot has great potential for other NASA-affiliated endeavors. This robot serves as a model for a search and rescue robot that could be utilized by the Stevens fire department; the robot could be sent in to drop a cell phone or an oxygen mask for a person trapped in a burning building. However, even in space, this robot could be used extensively. The robot could be deployed to drop off a beacon on another plane that could act as a marker for spacecraft to land. Or the robot could deliver medical aid & supplies to astronauts in space. Clearly, this project could be used in many applications in space.

Robot Design

Below is the sub system block diagram for the functions and design of the SRR.



In order to determine how to build the robot, an alternative design matrix was constructed in order to evaluate which design was the most efficient at achieving the objective of dropping the payload exactly 12 inches from the infrared beacon.

Search and Rescue Project Design Matrix

Design Parameters		All Combinations of Design Parameters comprising Conceprtual Designs							
		Ramp		**Conveyor belt**		**Arm & Funnel**		**Pulley**	
Lowering Mechanism		Inclined Plane		Belt		Arm & Funnel		Block-and-tackle	
Able to be powered by motor		Yes		Yes		Yes		Difficult	
Drop Cube within range		Difficult		Difficult		Yes		Yes	
Acceptance Criteria (as many as team likes)	**Weight (by %) of Acceptance Criteria (apply last)**	**Team Scoring of the importance of each Conceptual Design in meeting Acceptance**	**Weighted Score**	**Team Scoring of the importance of each Conceptual Design in meeting Acceptan**	**Weighted Score**	**Scoring of the importance of each Conceptual Design in meeting Acceptance Criteria**	**Weighted Score**	**Team Scoring of the importance of each Conceptual Design in meeting Acceptan**	**Weighted Score**
Aesthetics	15%	7	1.05	9	1.35	8	1.20	8	1.20
Accuracy	25%	7	1.75	7	1.75	9	2.25	7	1.75
Efficiency	25%	8	2.00	8	2.00	9	2.25	6	1.50
Weight(Light)	15%	7	1.05	7	1.05	8	1.20	8	1.20
Complexity	20%	6	1.20	9	1.80	9	1.80	8	1.60
	========		========		========		========		========
Total Percentage =	100%		7.05		7.95		8.70		7.25

Figure 1: Alternate Design Matrix

Four designs were conceived for a drop-off mechanism, although there could be numerous variations: a ramp, a conveyor belt, an arm and funnel, and a pulley. These designs were compared to the design parameters and acceptance criteria. The acceptance criteria were weighed by the team according to its importance to the particular design under consideration. The designs were evaluated on a scale of 1-10 (1 being the worst, 10 being the best) of how effectively they met a certain criterion. The scores were then cross multiplied and summed down, and the design with the highest score was chosen. This allowed us to determine mathematically which design was the most efficient. Since the third design, the arm & funnel had the highest score, it was the design chosen.

Design 1, the ramp would have involved the cube being pushed down an inclined plane. This design would require a motor to push an object that would send the cube down an inclined plane. It was a simple idea; but its lack of complexity and aesthetics made it unappealing.

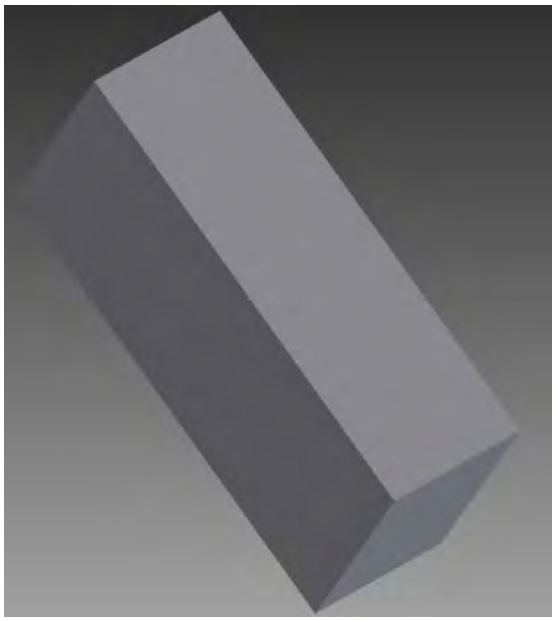


Figure 2: CAD model of the ramp

Design 2 a conveyor belt would have involved a motor controlling a belt that pushed the cube to the robot's edge. At that point the cube would fall down a slide. This idea was far more complex than the ramp design but it would put a lot of weight on the robot and greatly strain the motor controlling the conveyor belt and be an inaccurate drop.

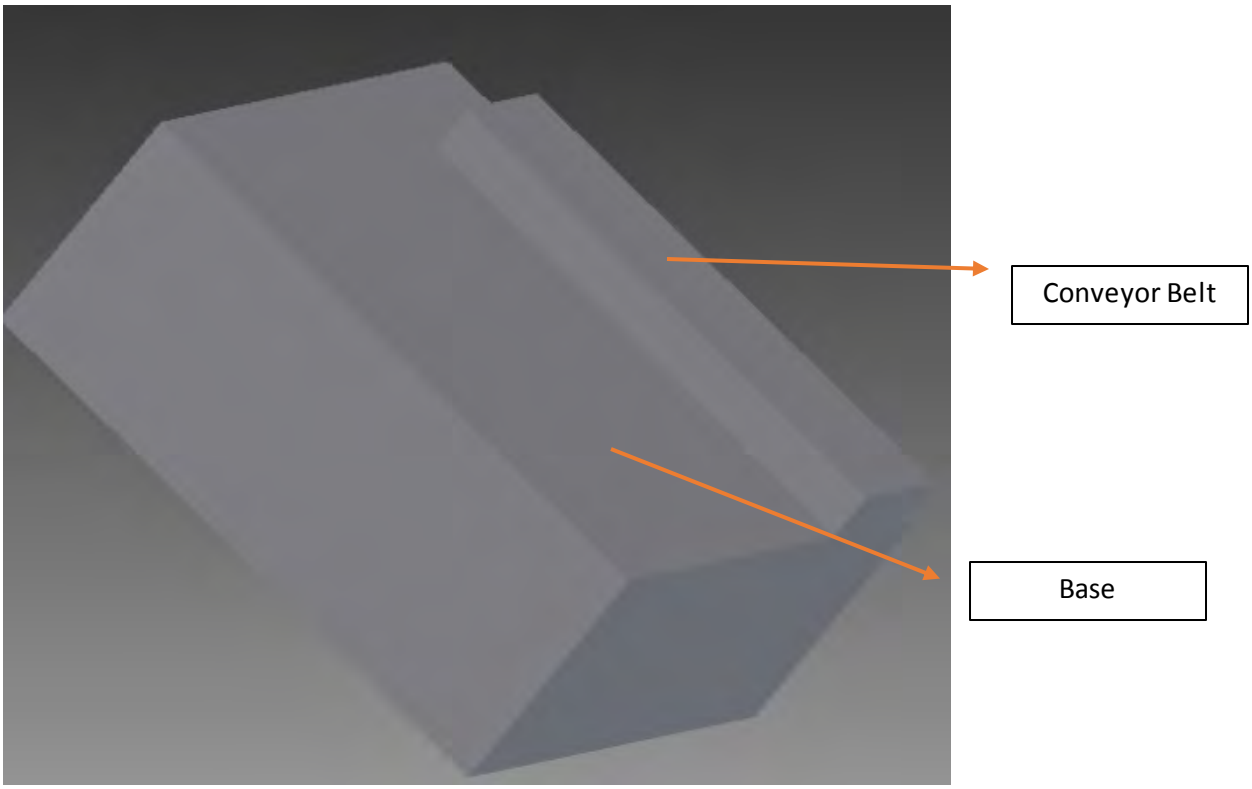


Figure 3: The Conveyor Belt design

Design 3, the arm and funnel was a simple but efficient design. A motor would control an arm that held the cube; the motor would then turn the arm and the cube would fall (by gravity) into the funnel. This concept wouldn't add too much weight to the robot, and wouldn't strain the robot. Nevertheless, it still maintained a degree of complexity and was aesthetically pleasing. This design proved that it could accomplish the objectives and achieved a high score on the design matrix.

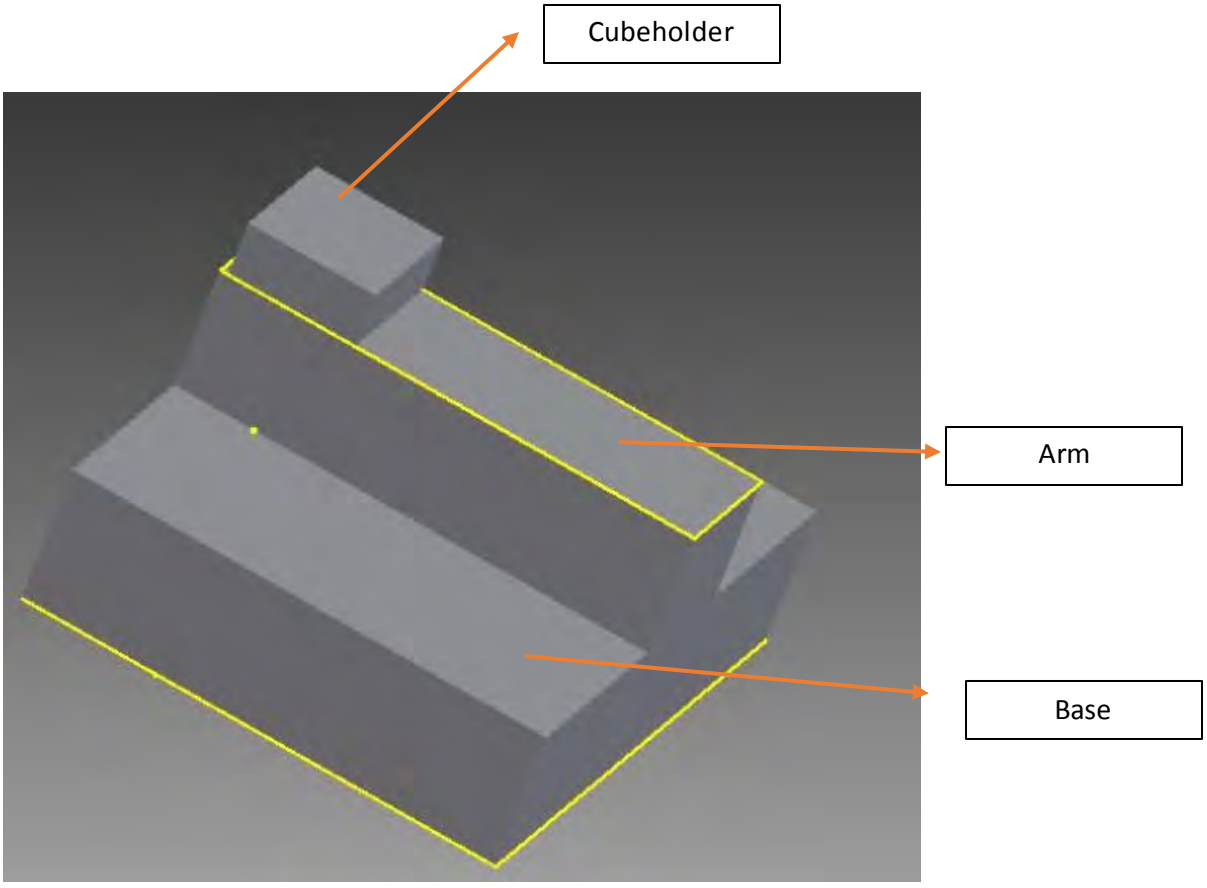


Figure 4: CAD model of Arm

The final design was a pulley. This idea would involve a block-and-tackle system. The cube would be on a ledge that was pointing downwards, but the cube would be prevented from falling by a block. When the robot reached 12 inches from the infrared beacon, a pulley would pull the block up and the cube would fall. In order to ensure a smooth & precise landing, a funnel or slide would be needed. Otherwise, the cube would be likely to bounce around and not land 12 inches away from the robot. Though this design also met objectives it was pretty complex and not very efficient. The pulley would likely run into many problems and would have to be re-done after every time the cube was dropped. Thus, the arm and funnel proved to be the most efficient design and was chosen.

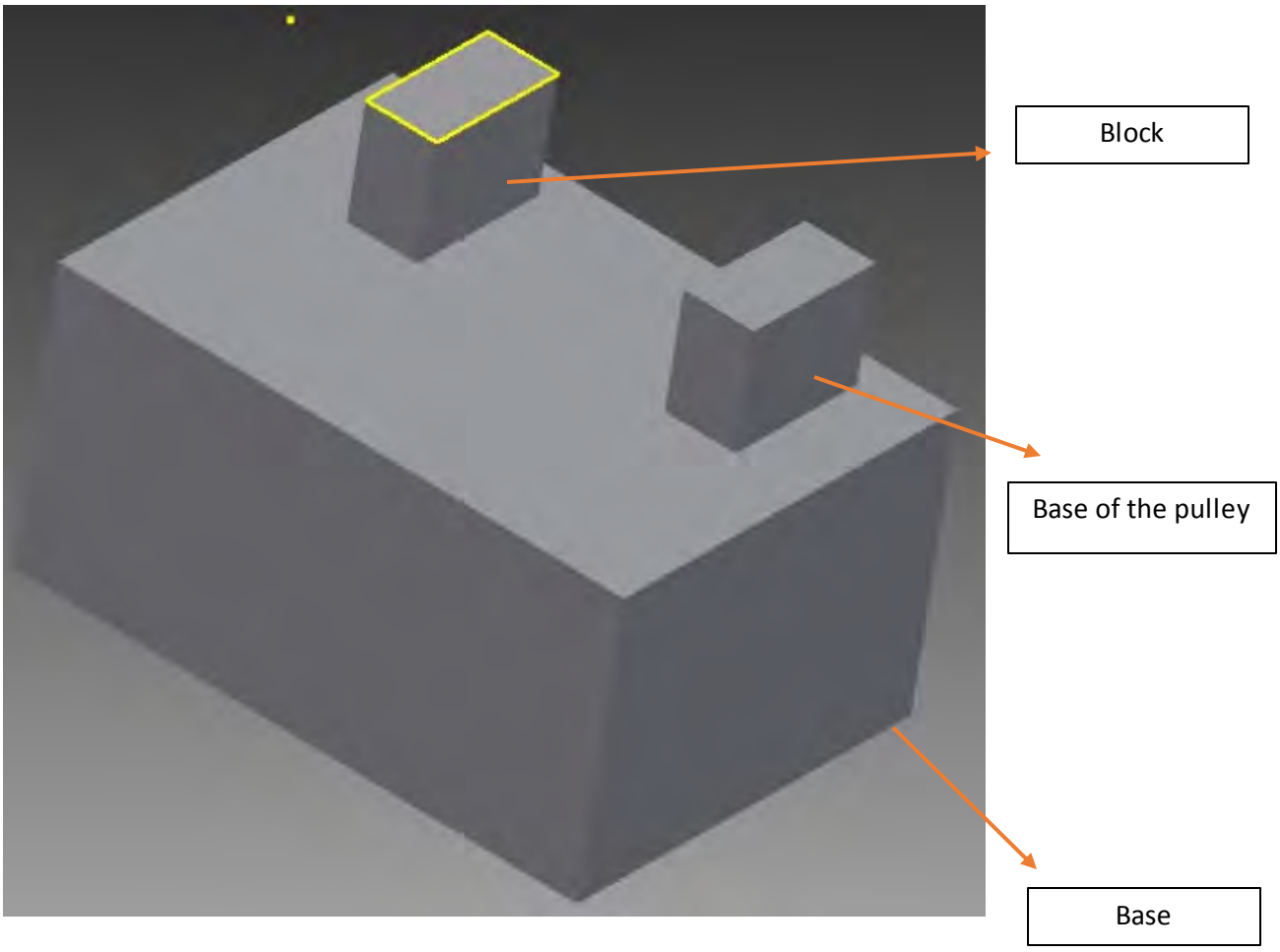


Figure 5: A CAD model of the pulley

Mechanical Engineering

After deciding the design, a material had to be chosen to construct the parts. Plastic was chosen. Plastic is moldable, sturdy, and aesthetically pleasing. It proved easy to cut, drill into, and shape. Additionally, this material displayed its sturdiness by holding the parts together and not falling apart even once. Finally, plastic added a nice finish to the project.

The mechanical engineering components include: the base, the supports, the cubeholder, the arm, and the funnel. The base is a 3" by 4.5" piece of plastic that sits atop the Arduino. It serves as a platform for the drop-off mechanism and is held up by the supports.

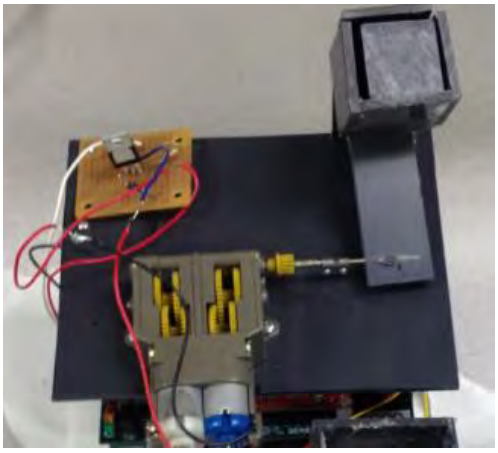


Figure 6: The base of the robot

The supports are 2" by 2" squares (one on each side of the robot). The supports hold the base up, and ensure a solid foundation for the drop-off mechanism. The cubeholder is a cube with each side being 1.2" long. It is close in dimension to the actual 1" cube. As a result, it holds the cube firmly but allows the cube to fall freely. This ensures the cube falls at the right time; not too early, not too late. The cubeholder is attached to the end of the arm.

Figure 7: The cubeholder with the cube

The arm is 3" by 1" piece of plastic that is connected via a shaft to a motor. This allows the arm to be rotated nearly 180 degrees. Additionally, the arm's end with cubeholder rests upon the ledge; the motor doesn't have to rotate the arm 180 degrees easing the motor's workload. The arm is rotated and drops the cube into the funnel which guides it to its destination.

Figure 8: Arm with the cubeholder. On the left of the arm is the shaft that connects it to the motor

The funnel is a 2" by 3" piece that is positioned just 1.5" above the ground. The cube is dropped into the funnel and the funnel's position close to the ground allows the cube to be dropped precisely at 12 inches.

Figure 9: The funnel

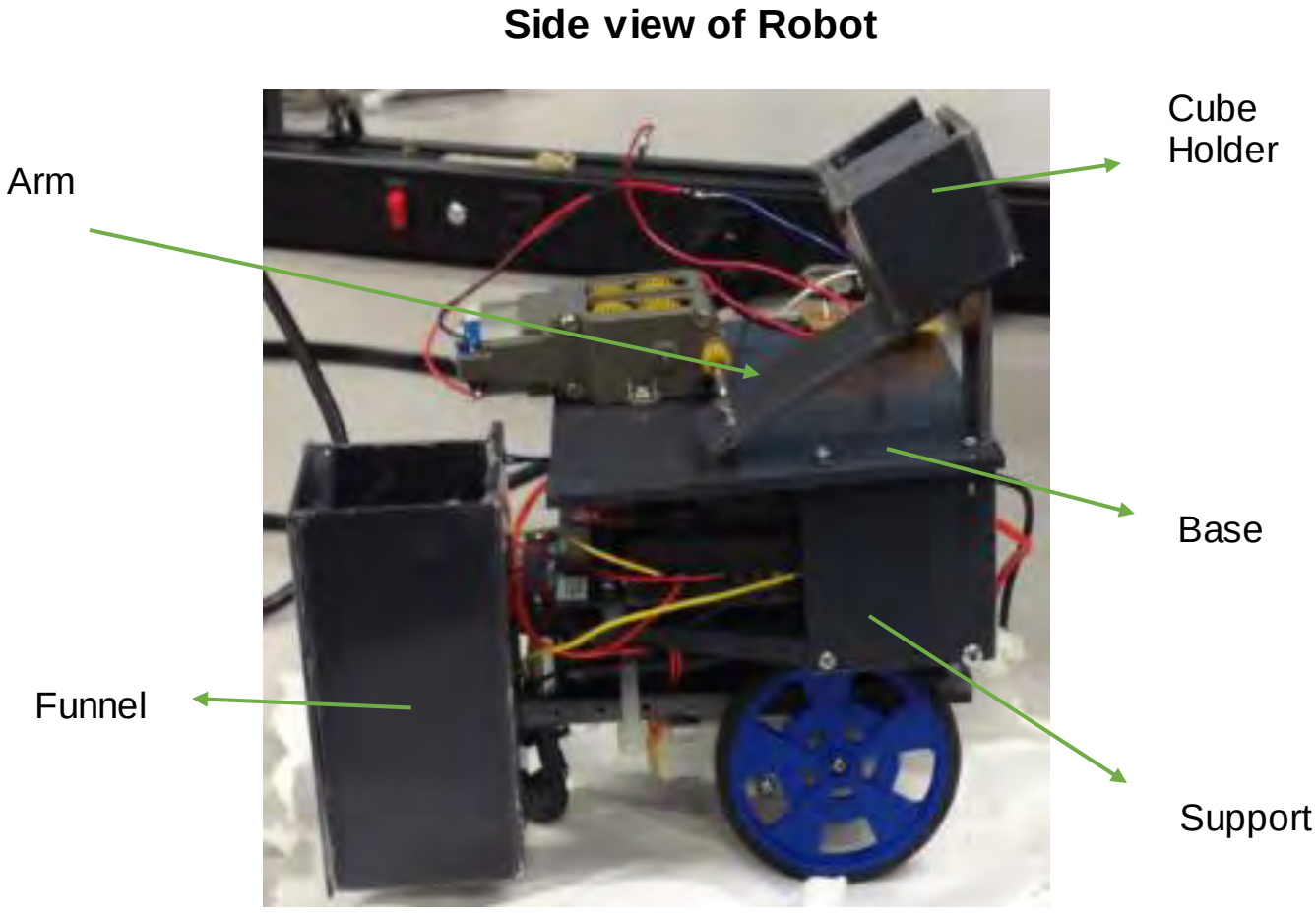


Figure 10: A side view of the robot with all the mechanical parts labelled

Electrical Engineering:

The team's SRR required a couple of sensors and external circuitry to fulfill its tasks. The sensors acted as variable inputs which allowed it to sense its environment. Stevens Institute of Technology customized the Arduino board to sustain the motor shield along with other functions necessary for the SRR to function correctly. The PIC board has a variety of I/O ports which allow for the wiring of sensors and other circuitry that is necessary. This board has analog ports, digital ports, and motor power ports. Analog ports are used to read a continuous voltage from the sensors while digital ports are used to turn the third motor on or off. The motor power ports are used to drive the two navigation motors. The PIC board has the capacity to sustain up to three motors. It is attached to a 9v battery along with two motors which are used to maneuver the two wheels attached to the robot chassis.

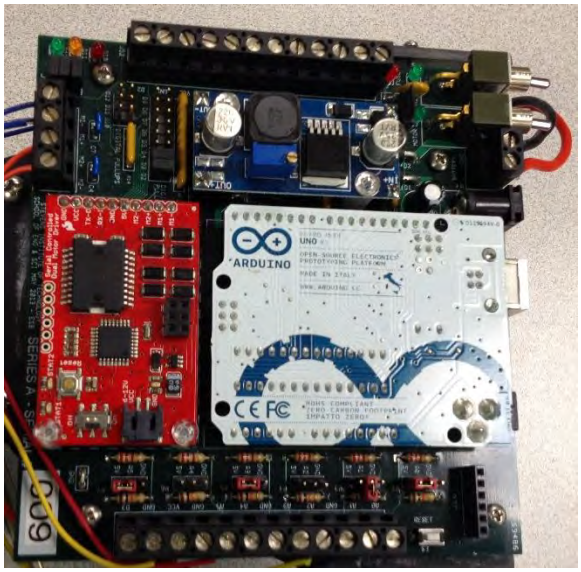


Figure 11: Arduino mounted on PIC board

The Arduino Uno acted as the brain of this robot. The Arduino processed the code and outputted instructions to the robot based on its input. This particular Arduino processed the sensor data along with the data that was coming in from the LabVIEW software. It constantly sent and received signals from both the software and the sensors attached to the PIC board on the robot.

The team used two main sensors to accomplish the objectives and reach the goal. The first sensor was the infrared sensor. This sensor was used to represent the detection of a human that

may be stuck under rubble or in a trapped building. The infrared sensor senses its surroundings by either emitting or detecting infrared radiation. These sensors can also be used to detect the heat of a human being or detect motion. Infrared light is not visible to the human eye and therefore cannot be seen without a phone. By using the camera app, you can see the light purplish-violet light being emitted from the source. When the Arduino receives the sensor data, the data is raw. This means that the value is not in a recognizable unit of measure; therefore, it must be converted by dividing by 10,000 to get a voltage value between 0-5 volts. The conversion is done in the LabVIEW Software.

Figure 12: Infrared

The second sensor that the team used was the proximity sensor. A proximity sensor emits an electromagnetic field or an electromagnetic beam of radiation and then monitors the changes in the field. Each sensor has a transmitter and receiver. The transmitter emits a signal which then bounces off the nearest physical object with which it comes in contact. The receiver then receives this signal and determines the distance between the sensor and the physical object. The Arduino then outputs this as raw data. The conversion from raw data to distance is non-linear and it needs further conversion. To alter the data to a value of inches, it is divided by 74, 938 and then the -0.8206^{th} root of that number is taken. This allowed the person driving the robot to see the exact positioning of the object in inches and stop at a particular distance away from the infrared emitter or human being. However for this particular project, the objective required the cube to be dropped exactly 12 inches away from the infrared beacon which represented a human.

Figure 13: Proximity Sensor

For the drop-off mechanism to function correctly, it required an external circuit so the Arduino and PIC board can support the third motor. This circuit consisted of a 1N4001 diode, 10k Ohms resistor, TIP121 NPN Transistor, and the third Lego motor. A schematic of the circuit is shown below.

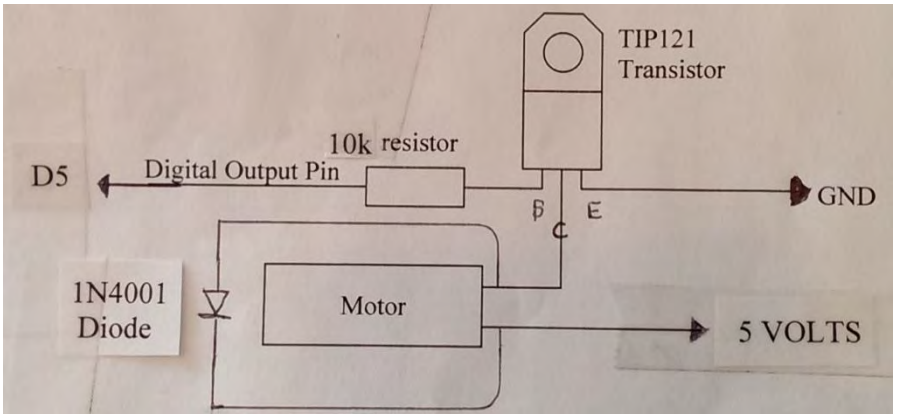


Figure 14: Schematic of Third motor circuit

The purpose of this circuit was to limit the current flow that goes to the motor so that the Arduino doesn't short out. The resistor limits the current and voltage, while the diode is put in parallel with the motor to turn it on and off. The image on the right is the third motor circuit attached to the base of the SRR. When 5 volts are sent to the motor, the motor turns on and spins the arm of the robot and drops the cube in to the funnel.

Figure 15: Third motor external circuit

Software Engineering:

LabVIEW (an acronym for Laboratory Virtual Instrument Engineering Workbench) is a system-design platform and development environment for a visual programming language from National Instruments. LabVIEW ties the creation of user interfaces (called front panels) into the development cycle. LabVIEW programs/subroutines are called virtual instruments (VIs). Each VI has three components: a block diagram, a front panel and a connector panel. The last is used to represent the VI in the block diagrams of other, calling VIs. The front panel is built using controls and indicators. Controls are inputs – they allow a user to supply information to the VI. Indicators are outputs – they indicate, or display, the results based on the inputs given to the VI.

The programming panel, which is called a block diagram, contains the graphical source code. All of the objects placed on the front panel will appear on the back panel as terminals. The block diagram also contains structures and functions which perform operations on controls and supply data to indicators. The structures and functions are found on the LabVIEW Functions Palette and can be placed on the block diagram. Collectively controls, indicators, structures and functions will be referred to as nodes. Nodes are connected to one another using wires – e.g. two controls and an indicator can be wired to the addition function so that the indicator displays the sum of the two controls. Thus a virtual instrument can either be run as a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the given node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.

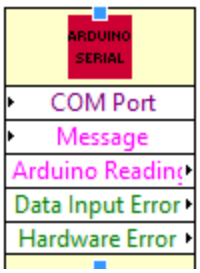


Figure 16: SubVI used in block diagram

The SubVI that was used in the block diagram was customized to interface with the Arduino on the PIC board. SubVI's are used to read the raw data from the analog and digital ports on the PIC board. Appendix A contains the block diagram for the SubVI that was used in the block diagram.

To drive the robot, the Logitech 3D Joystick Pro was used. This joystick contained 4 axes: X axis, Y axis, Z axis, & Z axis rotational. The X axis was used to move the SRR right and left, Y axis for up and down movement, and the Z axis was for when the joystick was in between the X and Y axis. Using the LabVIEW software, a sample block diagram was programmed to calibrate the joystick. This block diagram, shown below, consisted of a couple of components. First the LabVIEW software has to acquire the data and recognize the joystick. Then it takes the position of where the joystick is on the three axis and outputs it onto the front panel. After maneuvering the joystick in all four directions, the max value was 33,000 while the minimum was - 33,000 for each of the extremes on the joystick. These values were used in the main program/block diagram to determine the movement of the SRR.

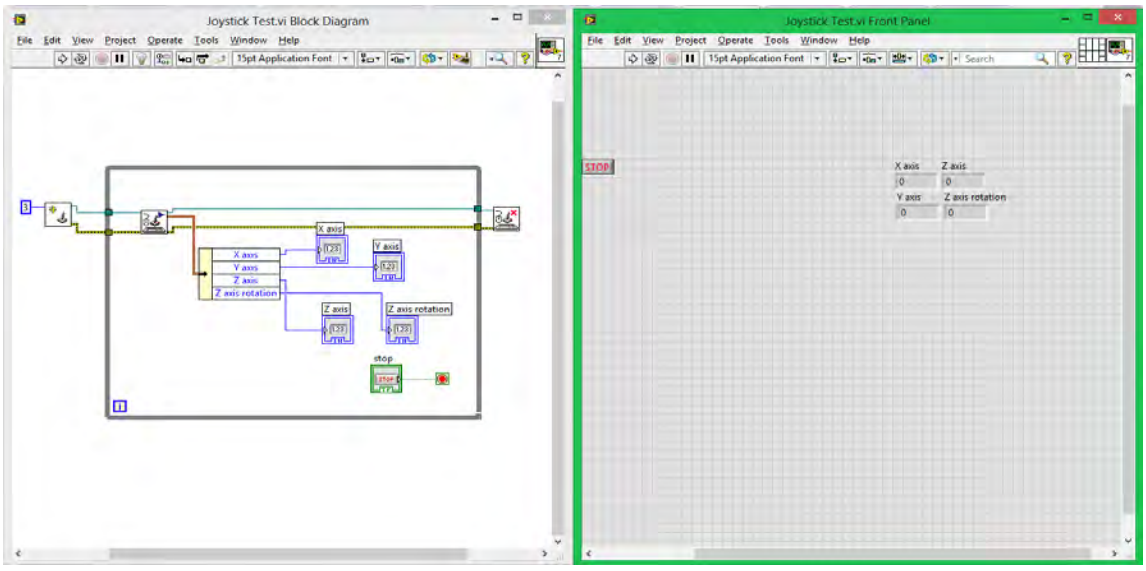


Figure 17: Joystick calibration block diagram

The main block diagram code, shown in Appendix A, consisted of three main portions. The first of these three portions included movement of the SRR. First, the joystick was initialized and connected to interface with the computer. Then, each of the axis values were outputted onto the front panel for debugging issues. Using the data interpreted from the team's sample block diagram for the joystick, a threshold of 30,000 was set. Then, using basic Boolean logic along with comparators, the direction of the SRR was determined by the outputted values, either 0 or 1. Each of the axis values were compared to the threshold, and then chosen based on the Boolean logic (AND & OR Gates). After that, based on which axis was greater, the output was fed into a case structure. This case structure included four different situations, up, down, right, or left. Once the proper case structure was executed, the motors were either turned on or off, and the direction was changed to comply with the direction of the robot. The motor speed is currently set at about 1.5 volts for precision.

The second part of the block diagram included the code for the sensor data. As mentioned above, two sensors were used to accomplish this task. Using the SubVI, signals coming in from the analog ports on the PIC board were analyzed by the software. This raw data was then received, processed by the block diagram, and outputted to the front panel for the user to see. The infrared data was divided by 10,000 to get a more accurate voltage reading. The proximity sensor raw data had to go through a series of conversions to appear as a value in inches, as described above. Along with that, a graph for each sensor was also put on the front panel. These graphs act as oscilloscopes in real life and output a graph of the sensor data received by the Arduino.

The last part of this block diagram was for the drop-off mechanism or package release. This included a simple true/false push button which is inputted into a true/false case structure. The case structure either turns the motor on or off. The motor is attached to a pulse width modulator

(PWM) digital pin on the PIC board. This allows for the motor speed to be altered for the users' desire.

The front panel or user interface for SRR consists of several components for the users' ease. As shown in appendix A, there are two indicators for "a data input error" and a "COM port error". Both these represent errors which may occur while connecting the SRR to a laptop. They are there for debugging warning lights to see if there are any problems with the input string or connection. The package release button is a simple true or false button which either turns the motor on or off. This is there for the users' benefit so that when the driver has reached twelve inches away from its target, they can drop off the marker or cube. Next, there are two graphs for the infrared and proximity sensor. The front panel shows the output of the Arduino and the converted data for each of the sensors. Along with that it outputs it onto a graph which depicts each of the values received vs the time. Last, there are output boxes for the joystick axis values. These values are used for debugging purposes. Overall, the front panel is designed for the users' ease and access to all the SRR functions. On the next page, there is a flowchart which goes through the high level algorithm when programming the SRR.

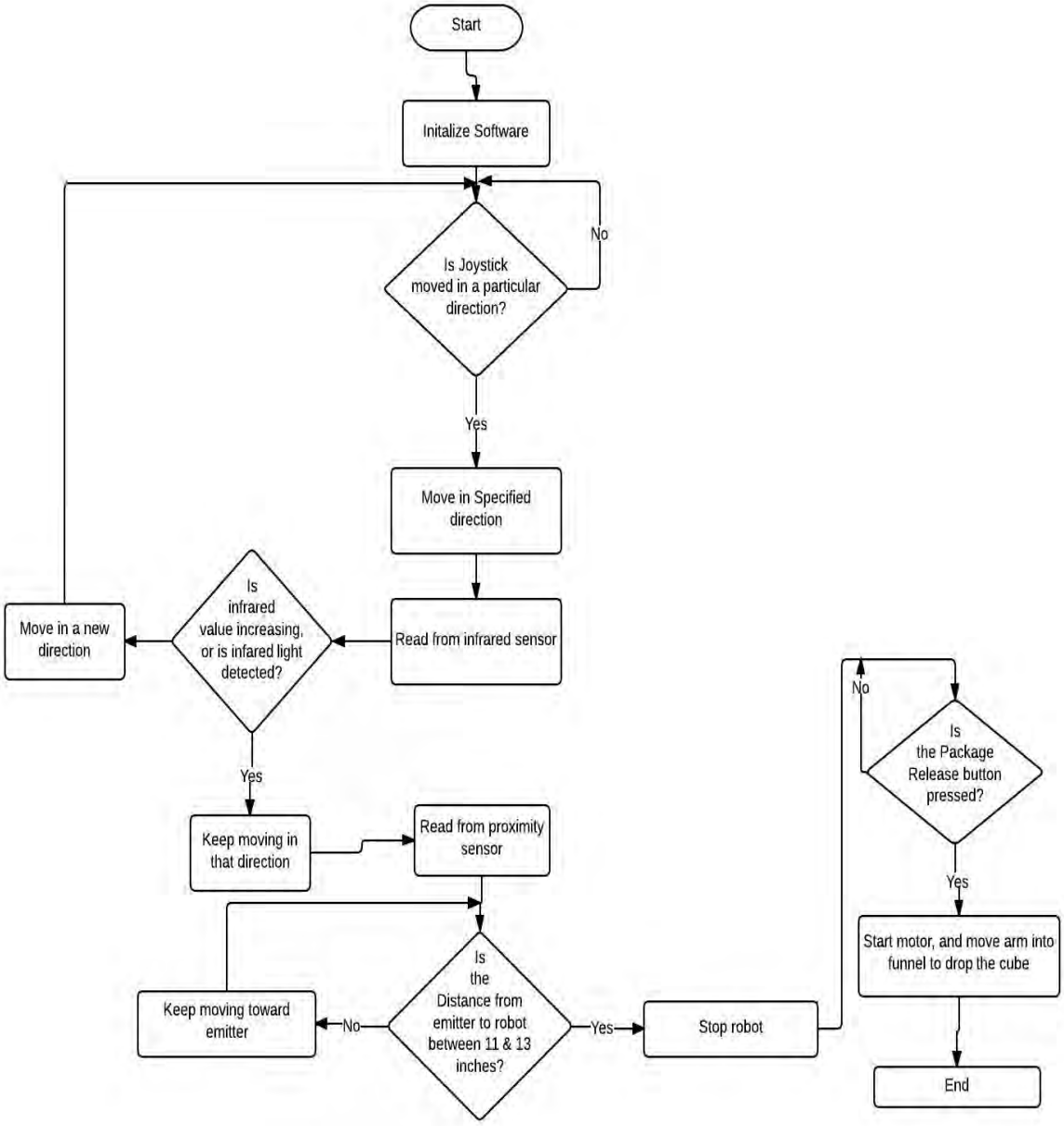


Figure 18: Flowchart for SRR Software

Testing:

To test the efficiency of the SRR, an arena at Stevens Institute of Technology was used. Three infrared light beacon emitters were set up around the arena behind or in front of a variety of the obstacles. The SRR was to be driven by the joystick and come close to the emitters. If the emitters were on, infrared was shown as a spike on front panel graphs, the user would start to look at the proximity sensor data. However, if the emitter was off, the SRR would continue to the next beacon, till it found one that was on and drop off the cube 11-13 inches away from the emitter. After a few trial and errors, the SRR was able to successfully locate the infrared emitter which was on 5/5 times. Below is an image of the arena the team set up along with an image of the infrared emitter.

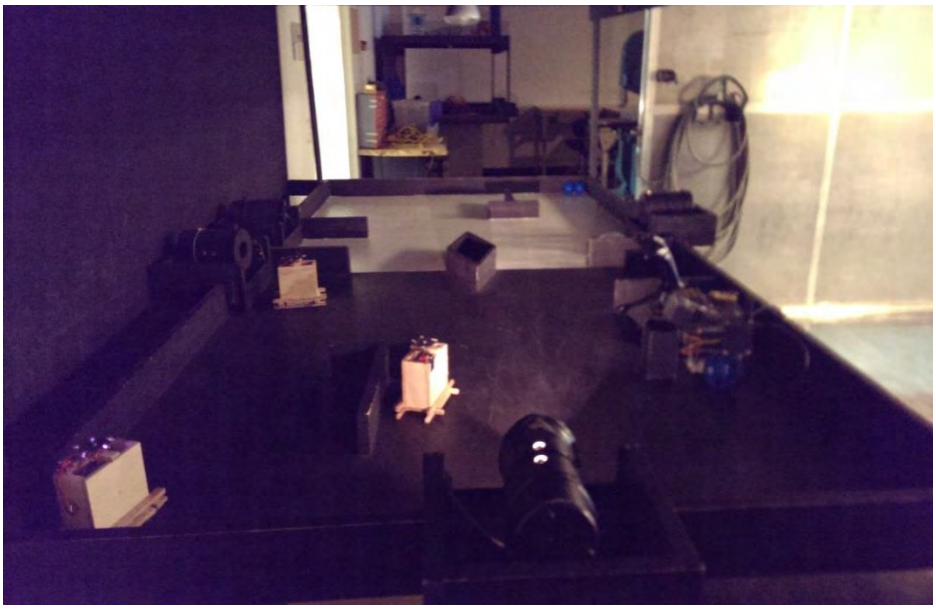


Figure 19: Arena with obstacles and infrared emitters

Figure 20: Infrared light emitter

Discussion:

Urban Search and Rescue (USAR) workers have 48 hours to find trapped survivors in a collapsed structure, otherwise the likelihood of finding victims still alive is nearly zero. Earthquake disaster mitigation requires rapid and efficient search and rescue of survivors. As recently seen in Turkey and Taiwan, the magnitude of the devastation of urban environments exceeds the available resources (USAR specialists, USAR dogs, and sensors) needed to rescue victims within the critical first 48 hours. Moreover, the mechanics of how large structures pancake often prevent heroic rescue workers from searching buildings due to the unacceptable personal risk from further collapse. Finally, and perhaps best addressed by the proposed work, both people and dogs are frequently too big to enter voids, limiting the search to no more than a few feet from the perimeter.

At the Carnegie Mellon Institute of Robotics, engineers are building a SRR called Serpentine. These serpentine robots have degrees of freedom most other robots don't have. They can move freely through narrow spaces without disturbing their surrounding environment. This unique form of movement allows for quicker rescues and can lead to many other types of similar robotics.

Figure 21: Serpentine SRR designed by CMU

The robotics department at Virginia Tech has developed a new type of locomotion for a SRR. This Amoeba like search and rescue robot has a torodial shape and actuator rings surrounding it. The motion of this robot is generated by contracting and expanding these rings. The entire contact with the surfaces for traction allows the robot to move through the environment with ease, going over uneven surfaces, and decreasing in diameter to fit through small spaces most robots wouldn't be able to access.

Figure 22: Amoeba like SRR

The teams search and rescue robot, though not as efficient as the robots described above, was able to complete its tasks. Though there might be a few things the team might have wanted to change about the SRR design to increase aesthetics and efficiency. First, superglue was used to hold parts of the plastic together, however that proved to be troublesome at times because it would come apart easily. Next time, the team would like to drill holes and properly attach it. Next, the movement of the robot could be a little bit more exact with a different algorithm. At the moment, the robot moves fine but the right turn is a little slower than the rest.

Financials:

Below is a list of materials that were used and the total cost for the completion of this project:

Item	Cost
Arduino Board	$30.00
Proximity Sensor	$11.00
Infrared Sensor	$1.00
Plastic	$5.00
Third Motor	$3.00
Chassis & Drive Motors	$10.00
Total Cost:	**$61.00**

Conclusion:

To conclude, the design for the SRR worked as efficiently as it could and completed its given objectives. It was able to successfully detect the infrared light, judge the distance between the robot and emitter, and drop the cube between 11-13 inches from the infrared emitter. The team's overall design was effective and easy to construct, also very aesthetically pleasing. The mechanical and electrical systems worked seamlessly with the LabVIEW software to create a small

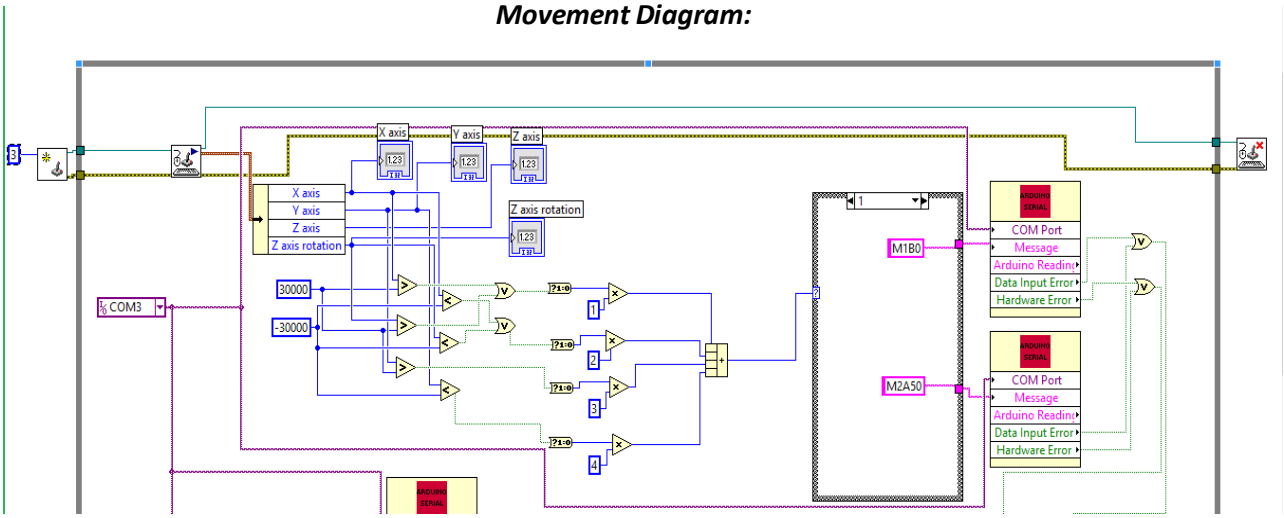
search and rescue robot. The team would like to thank NASA NYCRI for this opportunity as it produced promising results.

Appendices

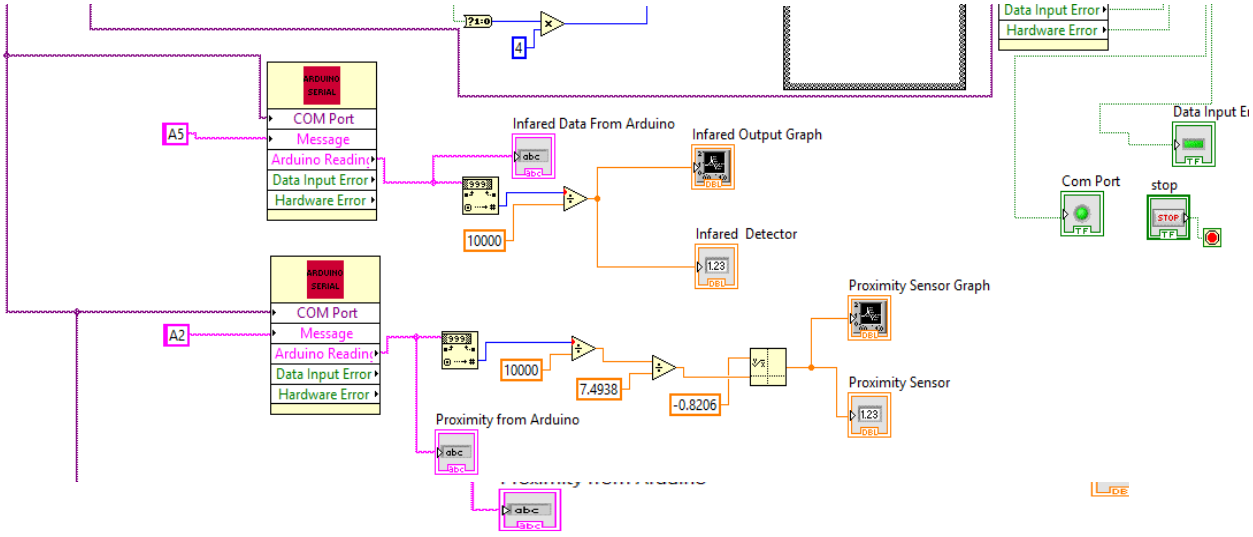
Appendix A:

Block Diagram from LabVIEW Software for SRR:

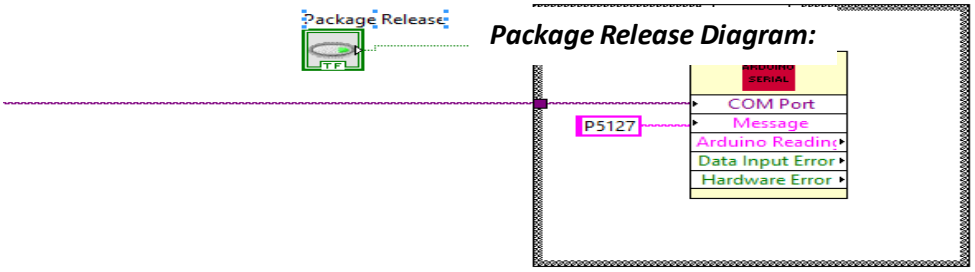
Movement Diagram:



Sensor Acquisition Diagram:

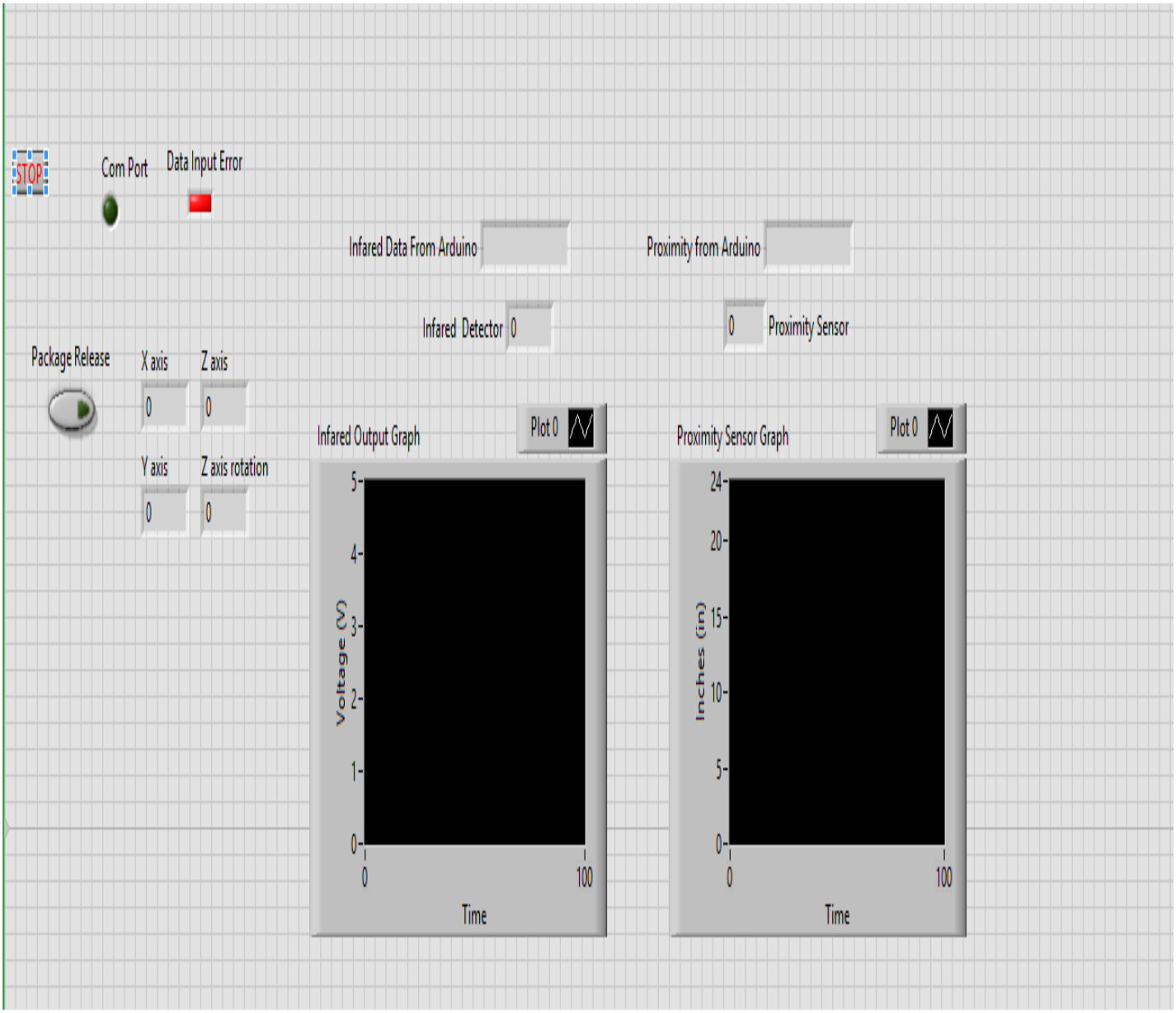


Package Release Diagram:



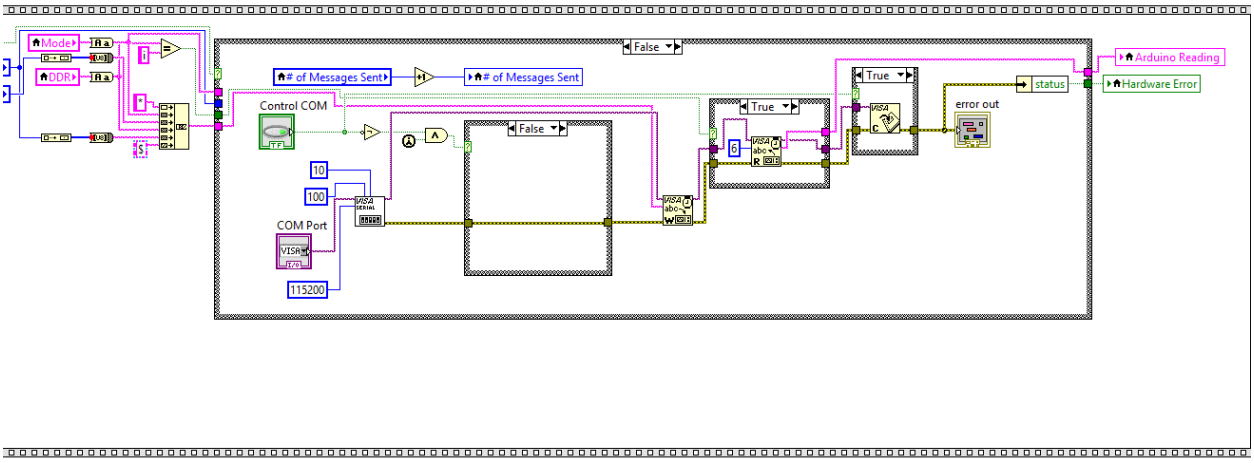
Appendix B:

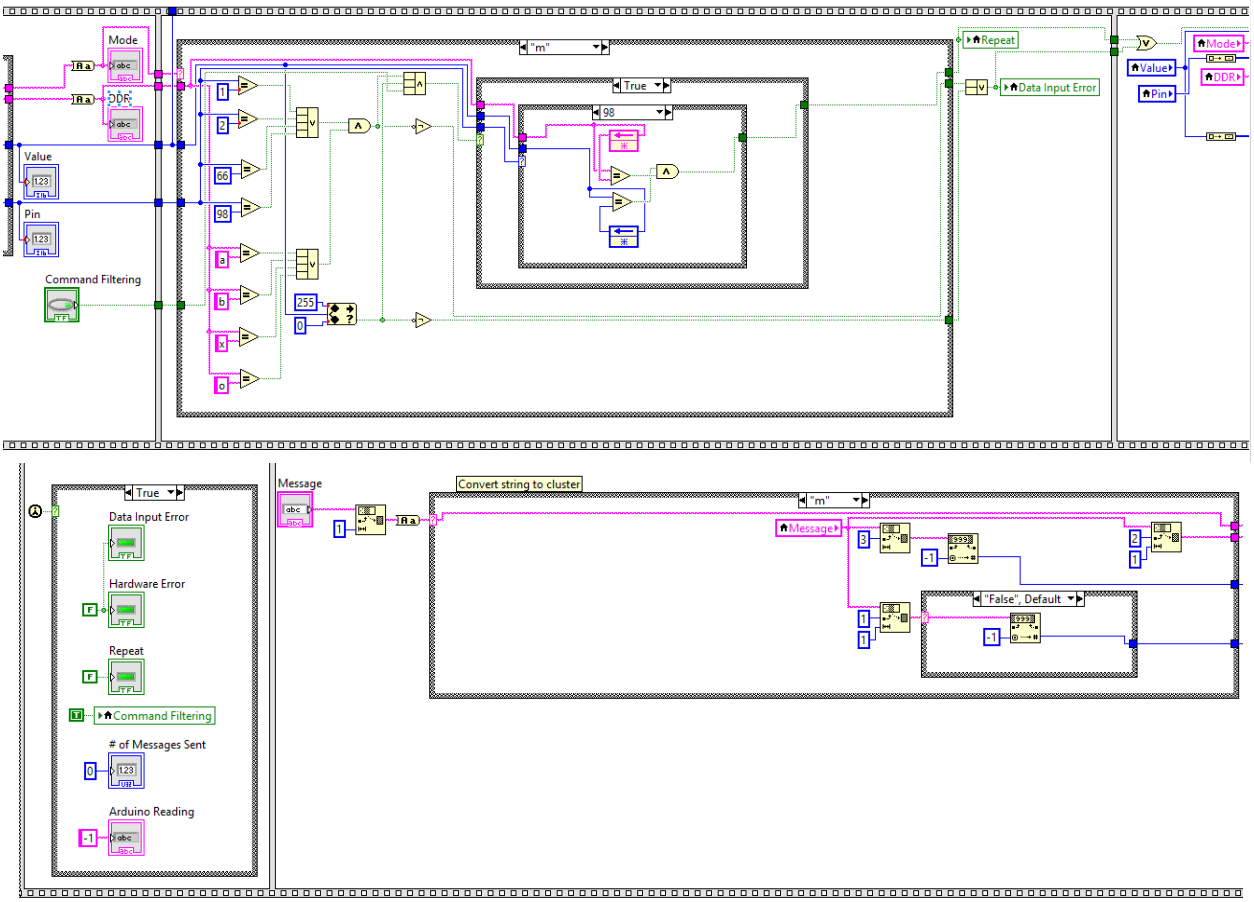
Front Panel or user interface for SRR:



Appendix C:

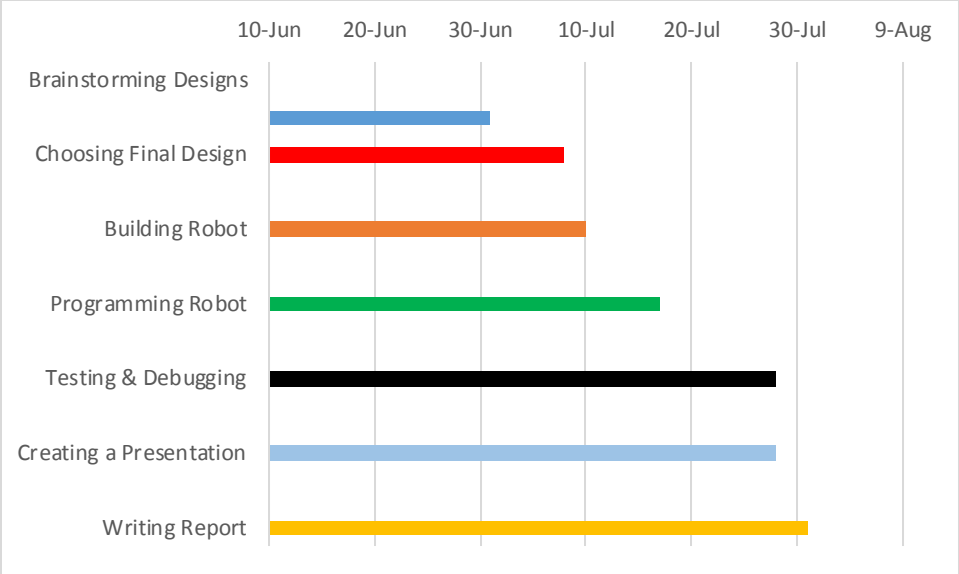
Block Diagram for SubVI used in programming USSR interface with Arduino:





Appendix D:

Gantt Chart of the Project



Sources:

Amoebalike Robots for Search and Rescue | MIT Technology Review. (n.d.). Retrieved from http://www.technologyreview.com/news/407603/amoebalike-robots-for-search-and-rescue/

Case Structure - LabVIEW 2012 Help - National Instruments. (n.d.). Retrieved from http://zone.ni.com/reference/en-XX/help/371361J-01/glang/case_structure/

Community: Need help with detecting the joystick, directX Working fine... - National Instruments. (n.d.). Retrieved from https://decibel.ni.com/content/thread/9122

How Can I Monitor a Joystick, Keyboard, or Mouse in LabVIEW? - National Instruments. (n.d.). Retrieved from http://digital.ni.com/public.nsf/allkb/CA411647F224787B86256DD000669EFE

Initialize Joystick VI - LabVIEW 2011 Help - National Instruments. (n.d.). Retrieved from http://zone.ni.com/reference/en-XX/help/371361H-01/glang/initialize_joystick/

Query Input Devices VI - LabVIEW 2011 Help - National Instruments. (n.d.). Retrieved from http://zone.ni.com/reference/en-XX/help/371361H-01/glang/query_input_devices/

Robotics Institute: Search and Rescue. (n.d.). Retrieved from https://www.ri.cmu.edu/research_project_detail.html?project_id=407&menu_id=261